Sandia National Laboratories

Exceptional service in the national interest

ME469: Advection Operators: Review

Stefan P. Domino[1,2]

[1] Computational Thermal and Fluid Mechanics, Sandia National Laboratories

[2] Institute for Computational and Mathematical Engineering, Stanford

SAND2018-4536 PE

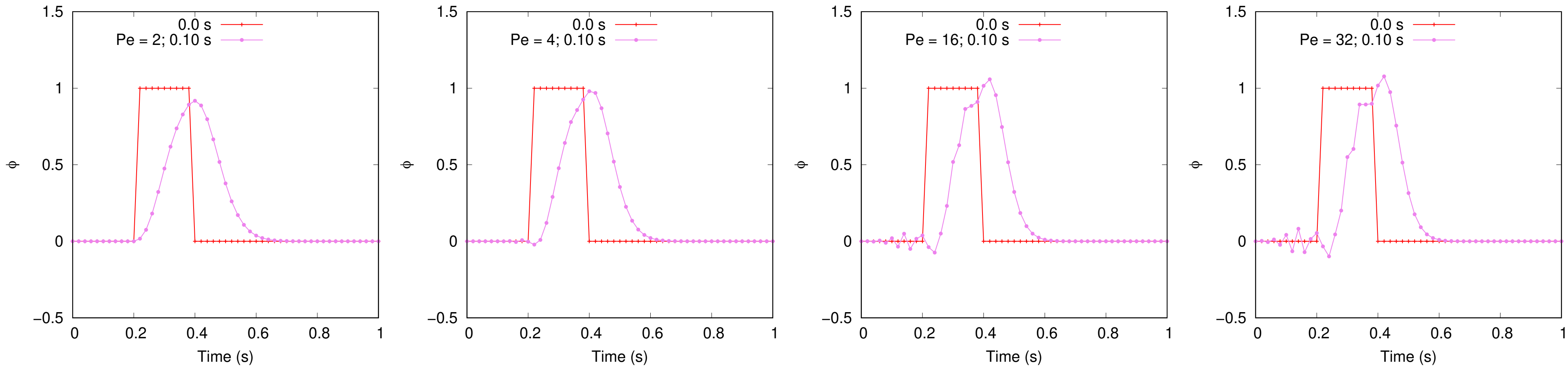
U.S. DEPARTMENT OF ENERGY

NNSA

Transient Advection/Diffusion: Step Function as Initial Condition Central, AKA Galerkin

Goal: Run our model equation with a variety of Peclet numbers using a step function as the initial condition

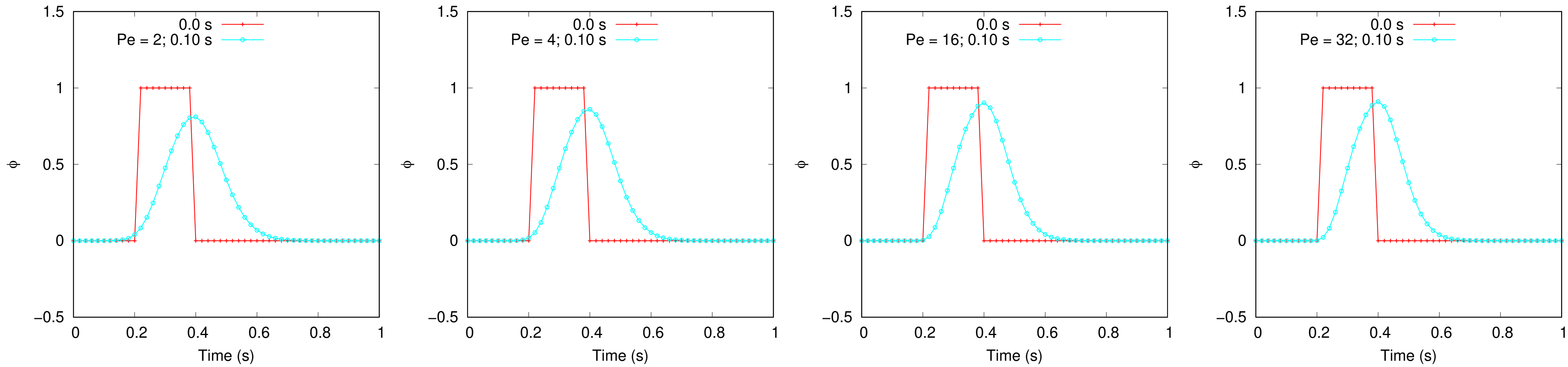
$$\int w \frac{\partial \rho u_j \phi}{\partial x_j} dV \approx \sum_{ip} (\rho u_j)_{ip} \phi_{ip} n_j dS \approx \sum_{ip} \dot{m}_{ip} \phi_{ip} \qquad \phi_{ip}^{CDS} = \sum_{n} N_n^{ip} \phi_n$$



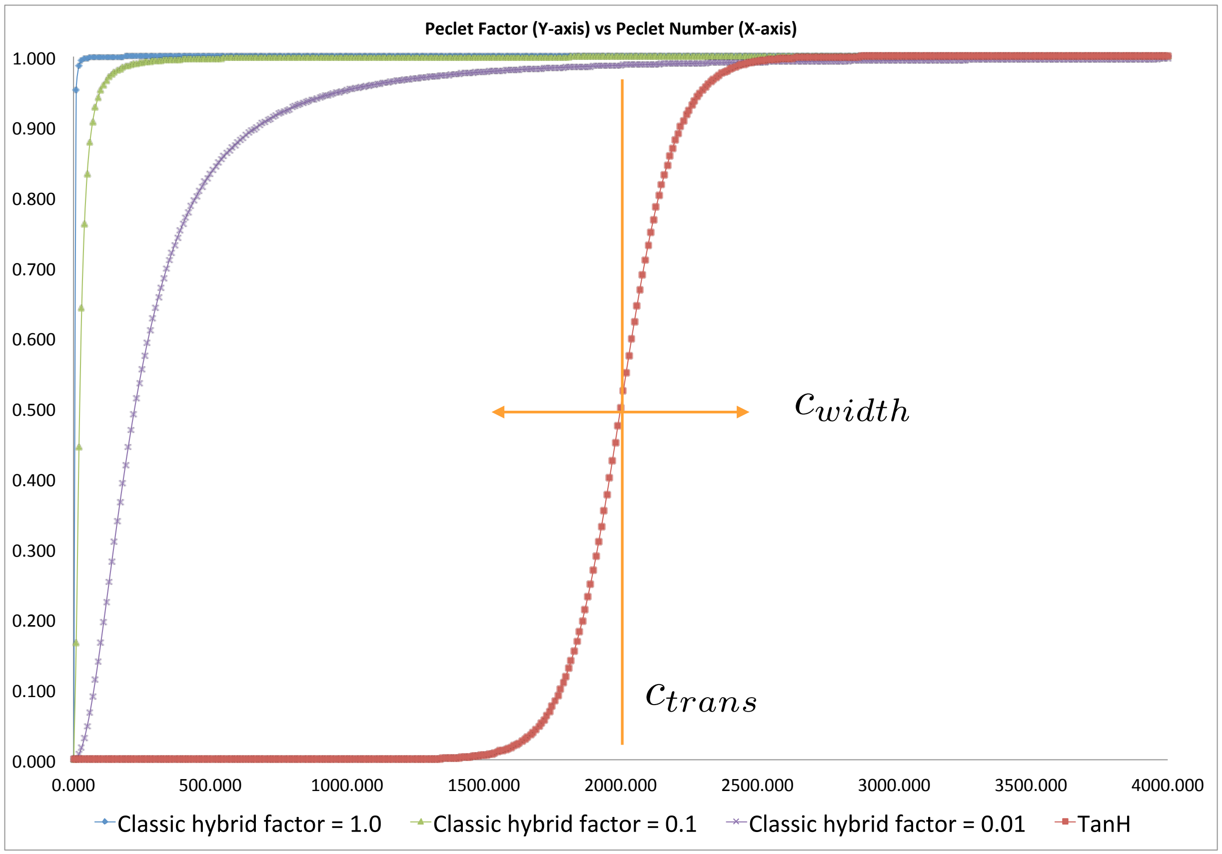
Transient Advection/Diffusion: Step Function as Initial Condition First-order Upwind

Goal: Run our model equation with a variety of Peclet numbers using a step function as the initial condition

$$\dot{m}_{ip}\phi_{ip}^{UPW} = \frac{\dot{m}+|\dot{m}|}{2}\phi_L + \frac{\dot{m}-|\dot{m}|}{2}\phi_R$$



Functional form for η – Linked to Peclet number, Pe
Many ad-hoc choices, however, a common physical approach is tanh



$$Pe = \frac{\rho U L}{\mu}$$

$$\eta = \frac{1}{2}\left[1 + tanh\left(\frac{Pe - c_{trans}}{c_{width}}\right)\right]$$

- peclet_function_form:
  velocity: tanh
  mixture_fraction: tanh

- peclet_function_tanh_transition:
  velocity: 5000.0
  mixture_fraction: 2.0

- peclet_function_tanh_width:
  velocity: 200.0
  mixture_fraction: 4.0

Kappa = 0 Method of Hirsh

Numerical Computation of Internal and External Flows, vol. 2, John Wiley & Sons, 1990.

- For $\kappa = 0$, recast as: (Algebra…..)

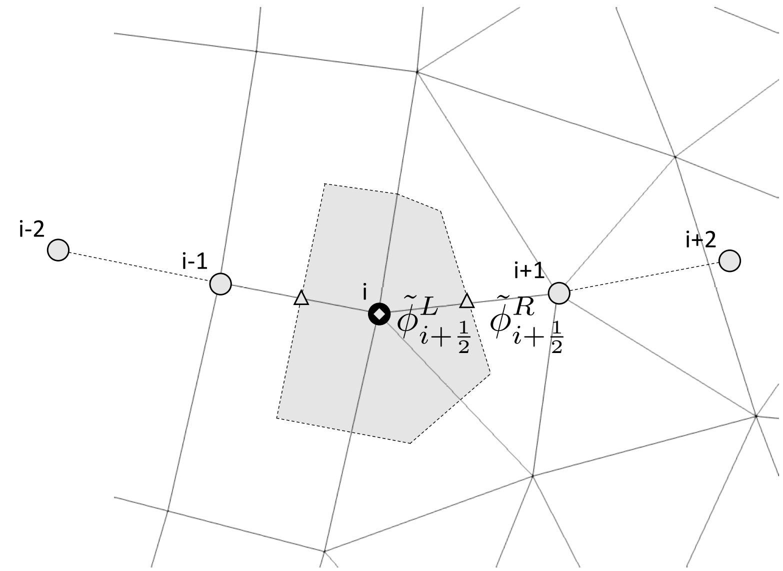
$$\tilde{\phi}^L_{i+\frac{1}{2}} = \phi_i + \Phi^L \Delta x_j^L G_j \phi_i,$$

$$\tilde{\phi}^R_{i+\frac{1}{2}} = \phi_{i+1} - \Phi^R \Delta x_j^R G_j \phi^{i+1}$$

Where,

$$\Delta x_j^L = x_j^{ip} - x_j^L,$$

$$\Delta x_j^R = x_j^R - x_j^{ip}$$



- Above, define a "limiter" function Φ^L, Φ^R that "senses" when the solution is smooth (tends towards unity) and when the solution is oscillatory (tends towards zero)
- G_j is the projected nodal gradient at each node (or cell-center) that is treated in a *deferred-correction* context, i.e., this quantity is lagged from the previous iteration
- So-called "gradient reconstruction" schemes
  - Reconstruct a higher-order stencil through extrapolation

Derived by substituting $\kappa = 0$, and using the projected nodal gradient definition – or – just by noting an extrapolation using a gradient

Assignment: Algebra!!!

Kappa = 0 Method of Hirsh: CVFEM

Numerical Computation of Internal and External Flows, vol. 2, John Wiley & Sons, 1990.

- For κ = 0, recast as: (Algebra.....)

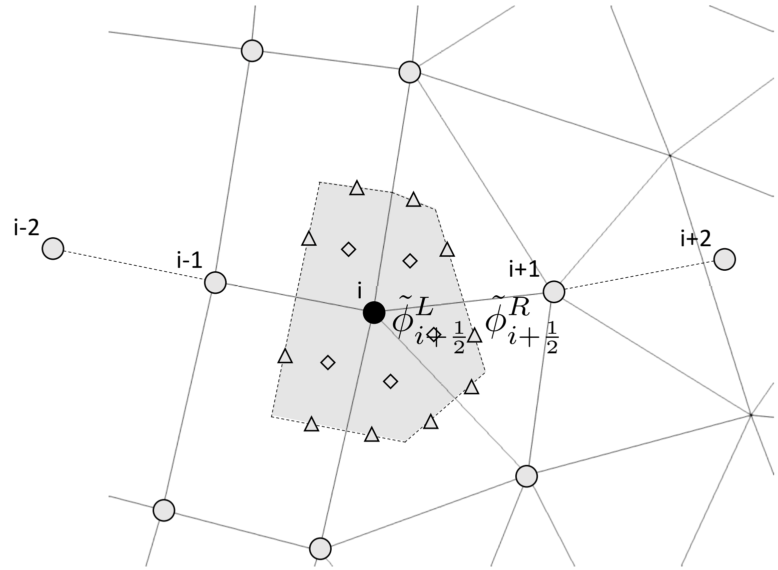
$$\tilde{\phi}^L_{i+\frac{1}{2}} = \phi_i + \Phi^L \Delta x^L_j G_j \phi_i,$$

$$\tilde{\phi}^R_{i+\frac{1}{2}} = \phi_{i+1} - \Phi^R \Delta x^R_j G_j \phi^{i+1}$$

Where,

$$\Delta x^L_j = x^{ip}_j - x^L_j,$$

$$\Delta x^R_j = x^R_j - x^{ip}_j$$



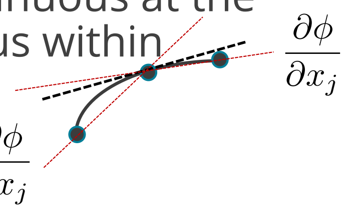
- Above, define a "limiter" function Φ^L, Φ^R that "senses" when the solution is smooth (tends towards unity) and when the solution is oscillatory (tends towards zero)
- G_j is the projected nodal gradient at each node (or cell-center) that is treated in a *deferred-correction* context, i.e., this quantity is lagged from the previous iteration
- So-called "gradient reconstruction" schemes
  - Reconstruct a higher-order stencil through extrapolation

CVFEM surface integration point can be along the edge, or at the standard SCS location with modified distance verctir

Assignment: Algebra!!!

Projected Nodal Gradient: Refresher

- Objective: We desire a nodal variable that represents the gradient of a scalar ϕ, $G_j\phi$
- We can view the nodal gradient as continuous at the nodes/DOF location, while discontinuous within element/control volume boundaries:



Let's minimize this difference (L_2):

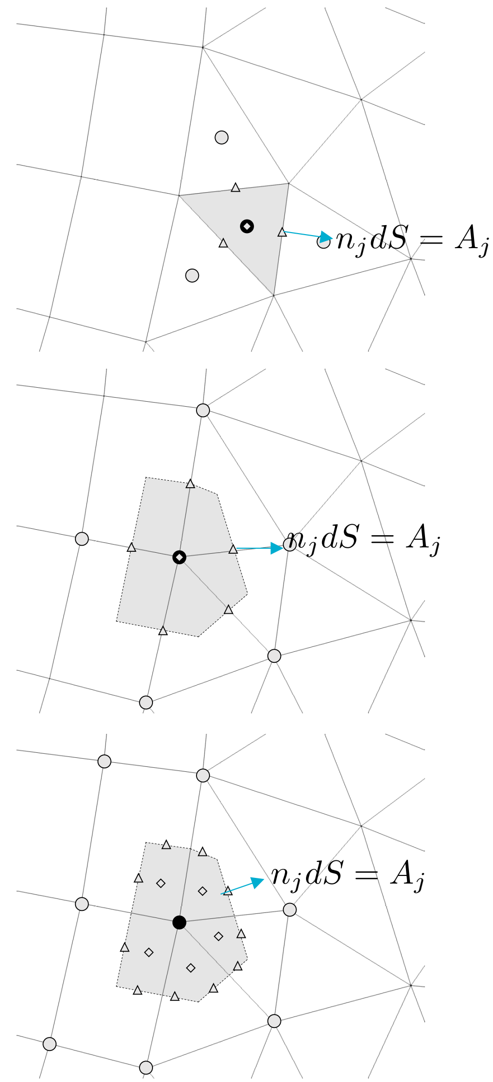
$$\int_\Omega \frac{1}{2}\left(\frac{\partial \phi}{\partial x_j} - G_j\phi\right)^2 d\Omega$$

by solving:

$$\int_\Omega w G_j \phi d\Omega = \int_\Gamma \phi n_j d\Gamma - \int_\Omega \frac{\partial w}{\partial x_j}\phi d\Omega$$

$$G_j\phi = \frac{\sum_{ip} \phi_{ip} n_j dS}{V}$$

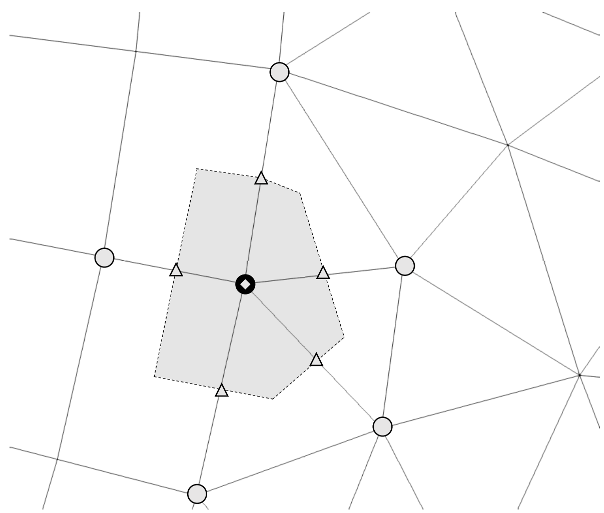
Lumped projected nodal gradient
(piecewise constant w)



Projected Nodal Gradient: Pseudo Code (Edge-based)

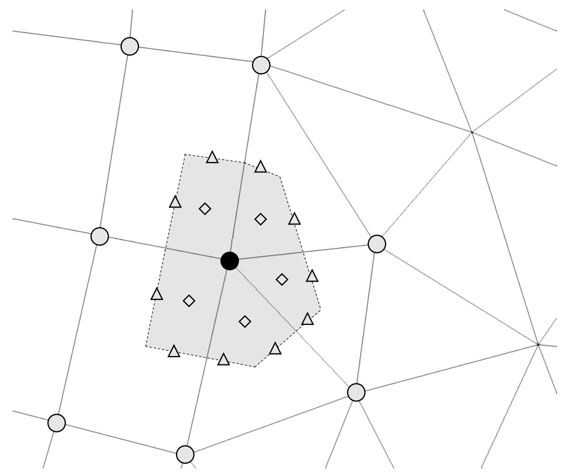
```
for ( stk::mesh::Bucket::size_type k = 0 ; k < length ; ++k ) {
     stk::mesh::Entity nodeL/nodeR = edge_node_rels[0]/edge_node_rels[1];
     const double qL = *stk::mesh::field_data( *scalarQ_, nodeL);
     const double qR = *stk::mesh::field_data( *scalarQ_, nodeR);
     const double qip = 0.5*(qL + qR);
     const double invVolL = 1.0/volL;
     const double invVolR = 1.0/volR;

     for ( int j = 0; j < nDim; ++j ) {
       const double aj = areaVector[k*nDim];
       gradQL[j] += aj*qip *invVolL;
       gradQR[j] -= aj*qip*invVolR;
     }
   }
 }
```

Rule: Area vector points from low to high global node ID

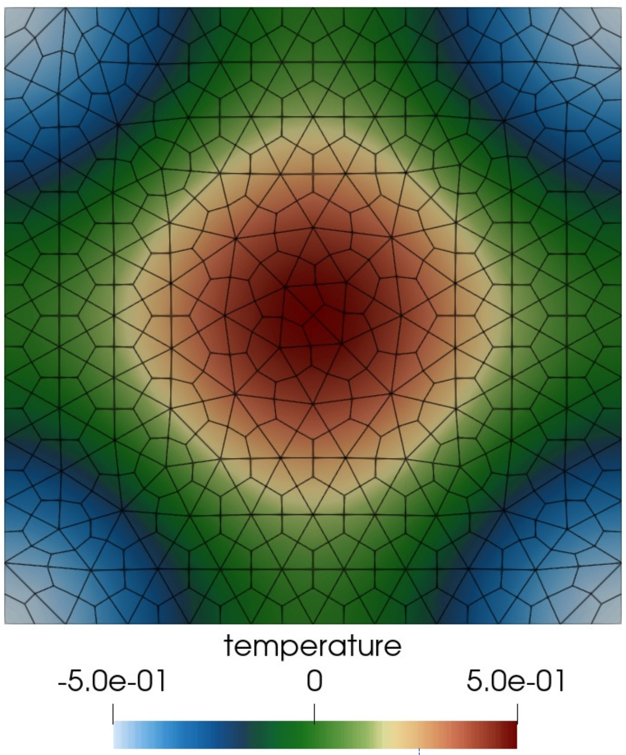


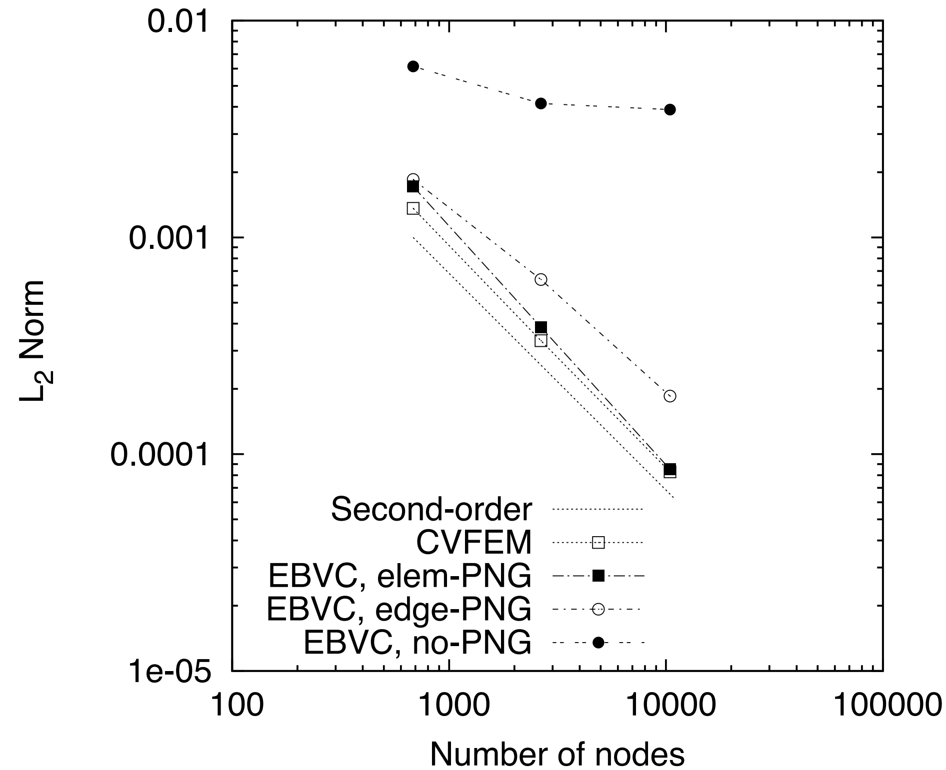
Projected Nodal Gradient: Pseudo Code (CVFEM)

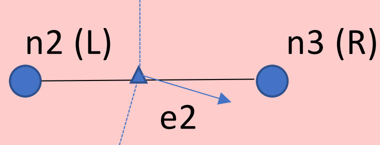
```
for ( int ip = 0; ip < numScsIp_; ++ip ) {

 // left and right nodes for this ip
 const int il = lrscv[2*ip];
 const int ir = lrscv[2*ip+1];

 double qIp = 0.0;
 const int offSet = ip*nodesPerElem_;
 for (int ic=0; ic < nodesPerElem;++ic ) {
   qIp += N[offSet+ic]*p_scalarQ[ic];
 }

 // assemble to il/ir
 for ( int j = 0; j < nDim_; ++j ) {
   double fac = qIp*areaVec[ip*nDim_+j];
   gradQL[j] += fac*inv_volL;
   gradQR[j] -= fac*inv_volR;
 }
}
```



For Instance, Verification of The Diffusion Operator







$$\frac{\partial \phi}{\partial x_j}\Big|ip = G_j^{ip}\phi + \left[(\phi_R - \phi_L) - G_l^{ip}\phi\Delta x_l\right]\frac{A_j^{ip}}{A_k \Delta x_k}$$

Solver Nuance

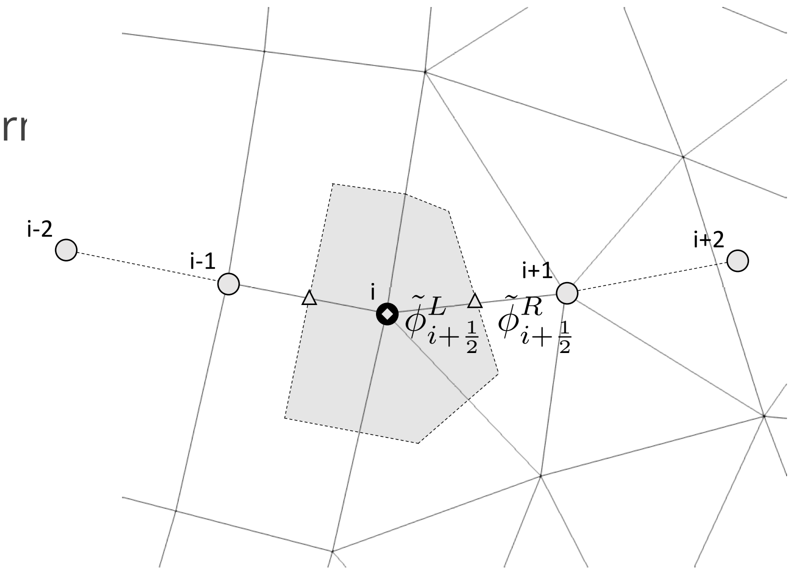
Recall, we like to solve our systems in residual- or delta-for

$$M\Delta x^{k+1} = -R^k = b - Ax^k$$

$$LHS_R \approx \frac{\dot{m} - |\dot{m}|}{2} \qquad LHS_L \approx \frac{\dot{m} + |\dot{m}|}{2}$$

$$\tilde{\phi}^L_{i+\frac{1}{2}} = \phi_i^{k+1} + \Phi^L \Delta x_j^L G_j^k \phi_i,$$

$$\tilde{\phi}^R_{i+\frac{1}{2}} = \phi_{i+1}^{k+1} - \Phi^R \Delta x_j^R G_j^k \phi^{i+1}$$



The Idealized Stencil Set

With Nalu input file specifications

$i-2$	$i-1$	i	$i+1$	$i+2$	α	α_{upw}
0	$-\frac{1}{2}$	0	$+\frac{1}{2}$	0	0	n/a
$+\frac{1}{8}$	$-\frac{6}{8}$	0	$+\frac{6}{8}$	$-\frac{1}{8}$	$\frac{1}{2}$	n/a
$+\frac{1}{12}$	$-\frac{8}{12}$	0	$+\frac{8}{12}$	$-\frac{1}{12}$	$\frac{2}{3}$	n/a
$+\frac{1}{4}$	$-\frac{5}{4}$	$+\frac{3}{4}$	$+\frac{1}{4}$	0	$\dot{m} > 0$	1
0	$-\frac{1}{4}$	$-\frac{3}{4}$	$+\frac{5}{4}$	$-\frac{1}{4}$	$\dot{m} < 0$	1
$+\frac{1}{6}$	$-\frac{6}{6}$	$+\frac{3}{6}$	$+\frac{2}{6}$	0	$\dot{m} > 0$	$\frac{1}{2}$
0	$-\frac{2}{6}$	$-\frac{3}{6}$	$+\frac{6}{6}$	$-\frac{1}{6}$	$\dot{m} < 0$	$\frac{1}{2}$

- alpha_upw:
  velocity: 1.0

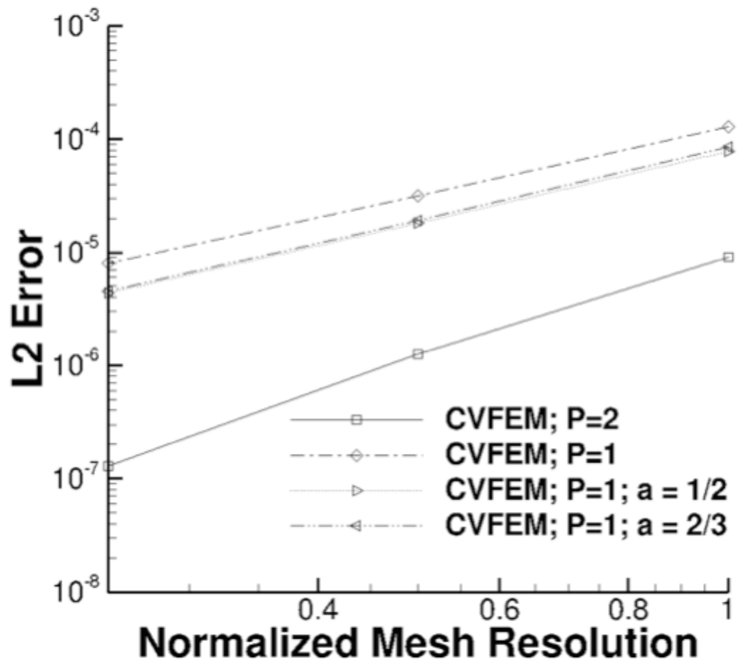
- alpha:
  velocity: 1.0

- upw_factor:
  velocity: 1.0 (zero reverts to first-order)

- limiter:
  velocity: [yes/no]

Pseudo 4th order Verification Results

Verification using Central (linear and quadratic) compared to pseudo 4th order



Lower error, however, formally second-order accurate